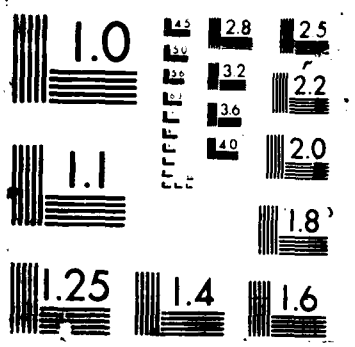
AD-A187 615 LATENT FAILURES AND COVERAGE IN FAULT-TOLERANT SYSTEMS 1/1
A VLSI CMOS CIRCUI..(U) STANFORD UNIV CA CENTER FOR
RELIABLE COMPUTING H H AMER ET AL. DEC 86 CRC-TR-86-18
UNCLASSIFIED MDA903-83-C-0335 F/G 9/1 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

Center for Reliable Computing

CONTRACT: MDA 903 83 C 0335

Preprints

Amer, H.H., and E.J. McCluskey, "Latent Failures and Coverage in Fault-Tolerant Systems," IEEE Pheonix Conference on Computers and Communications, Scottsdale, AZ, Feb. 25-27, 1987.

Liu, D.L., and E.J. McCluskey, "A VLSI CMOS Circuit Design Technique to Aid Test Generation," IEEE Pheonix Conference on Computers and Communications, Scottsdale, AZ, Feb. 25-27, 1987.

CRC Technical Report No. 86-18

(CSL TN No. 86-307)

December 1986

CENTER FOR RELIABLE COMPUTING
Computer Systems Laboratory
Electrical Engineering and Computer Science Departments
Stanford University, Stanford, California

ABSTRACT

This Technical Report contains preprints of papers to be presented at the IEEE Pheonix Conference on Computers and Communications, Scottsdale, AZ, Feb. 25-27, 1987.

This work was supported in part by the National Science Foundation under Grant No. NSF DCR8200129 , the Defense Advanced Research Projects Agency (DARPA) under Grant No. MDA. 903-83-C-0335, and the Egyptian Government.

TABLE OF CONTENTS

LATENT FAILURES AND COVERAGE IN FAULT-TOLERANT SYSTEMS

Hassanein H. Amer and Edward J. McCluskey

Center for Reliable Computing
Computer Systems Laboratory, Stanford University
Stanford, CA 94305, USA

ABSTRACT

A method is presented to include the effects of latent failures in the coverage parameter calculation for fault-tolerant systems. Programs for estimating the reliability of fault-tolerant systems do not explicitly take into account the effect of latent failures in the hardware recovery mechanism. This paper shows how to incorporate these failures in the fault-handling (coverage) model of CARE III. The method presented produces an excellent estimate of the reliability of the fault-tolerant system when incorporated into CARE III.

1 INTRODUCTION

Analytical models have been developed to estimate the reliability of computer systems. These models can be applied to a large class of fault-tolerant systems [Bavuso 84] [Bridgman 84]. The user must calculate the required parameters for these models. One of these parameters is the coverage, the conditional probability of successful error recovery given that an error has occurred [Borgerson 75] [Bouricius 71]. Error recovery consists of error detection, isolation and system reconfiguration. The sensitivity of reliability to a small error in the coverage estimation is well known [Arnold 72]. The reliability of the hardware responsible for the error recovery must be taken into account [Losq 75].

CARE III is a well-known automated reliability model. It has a separate model for the coverage in which it is assumed that the isolation of a detected error and the recovery from it will always be successful [Bavuso 84] [Trivedi 81]. Thus, the coverage model in CARE III takes only error detection into account. Furthermore, CARE III does not explicitly model latent faults in hardware recovery mechanisms. Latent faults are faults that will not generate errors until another fault occurs [Siewiorek 82].

In [Amer 86a], a stand-by spare system was designed. The system had a hardware recovery mechanism. The faults were analyzed and classified to point out the difficulties encountered in the calculation of the coverage parameters. The fault-handling (coverage) model of CARE III had to be used twice: once for the latent faults in the recovery mechanism and once for the other faults in the system (active module, stand-by module and recovery mechanism).

In this paper, the circuit presented in [Amer 86a] is modified. The hardware recovery mechanism systematically exercises each of the two modules i.e. both modules are interchangeably connected to the system bus. The faults are classified and it is shown how to include the faults in the hardware recovery mechanism in the coverage model of CARE III. As in [Amer 86a], the reliability estimate obtained from CARE III is conservative when compared to an estimate obtained from a model specific to the system.

It is then shown that a simplified version of the CARE III coverage model produces a reliability estimate that is identical to that obtained from a model specific to the system under study. It is conjectured that the computer roudoff error is responsible for the difference between the estimates obtained from the original and simplified coverage models. Since the system being analyzed is very simple, the interarrival time between failures will be much longer than the time needed to recover from a failure. Therefore, it is reasonable to neglect the recovery time in the fault-handling (coverage) model of CARE III.

In Sec. 2, the system is described. The faults are divided into different classes and some examples are given to show the effect of the faults in the recovery mechanism on the reliability of the system. Only permanent faults are modeled. In Sec. 3, a reliability model is developed that is specific to the system under study. In Sec. 4, the coverage parameters and the reliability of the system are calculated using the models in CARE III. Special attention is given to the calculation of the coverage parameter "c" or the probability of a failure not being lethal to the system [Bavuso 84]. The reliability calculated using CARE III is then compared to a reliability prediction obtained from the Markov model described in Sec. 3. It is shown that the CARE III models produce a conservative estimate of reliability. In Sec. 5, the fault-handling (coverage) model in CARE III is modified to eliminate the parameters and states involving recovery time. The reliability estimate obtained using the simplified model is identical to the one obtained when using the specific model.

2 THE FAULT-TOLERANT SYSTEM

The function of the system is equivalent to that of a 2-input OR gate. Figure 1 shows the fault-tolerant system. It has two inputs: Bus-1 and Bus-2. There are two identical outputs: Bus-out-1 and Bus-out-2. The system consists of two identical modules (X and Y). Module X consists of an OR gate and a NAND gate with two inverted inputs. In the fault-free situation, the outputs of the OR and NAND gates are identical. The NAND

gate is redundant and is used for error detection within the module. An EXCLUSIVE-OR gate (the detector) compares the outputs of the OR and NAND gates to detect any error. The output of the OR and NAND gates are connected to the bus through two buffers with 3-state outputs. Module Y is identical to module X. The switch (in each module) is implemented by an OR gate, a D latch and two gates to control the buffers. Initially, the D latches are reset (Q_x=0 and Q_y=0). The buffers are controlled by the states of the two latches and by the logic value of the signal on Bus-1 (one of the inputs). In the fault free situation, Q_x and Q_y are both equal to 0. Module X will be connected to the bus when the signal on Bus-1 has a logic value 0 and module Y will be connected to the bus when the value of the signal on Bus-1 is a logic 1. This way, modules X and Y will be systematically exercised (or "flexed"). If an error is detected in one of the two modules, this module is permanently disconnected from the system bus. If the EXCLUSIVE-OR gate in module X, for example, detects a discrepancy between the outputs of the OR and NAND gates, the latch toggles, module X is permanently disconnected from the bus and module Y is permanently connected to it.

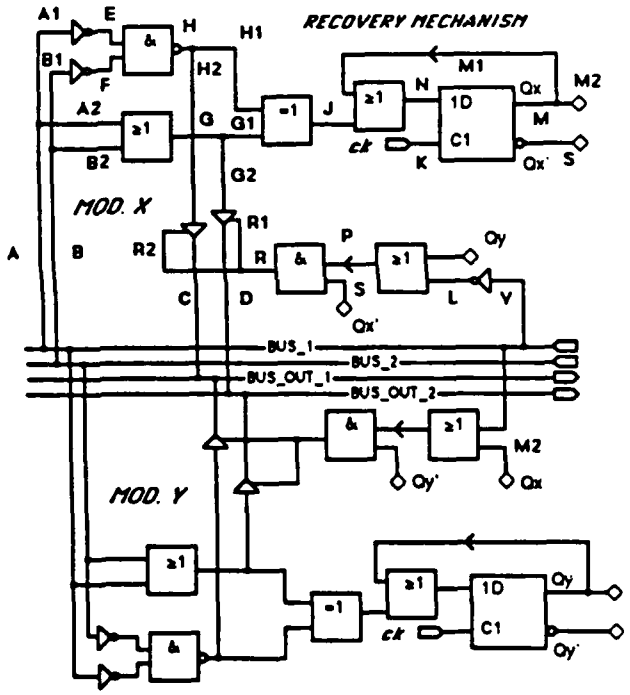


Fig. 1 Fault-tolerant System

For this system, the recovery mechanism is more complex than the redundant modules that implement the OR function. A more realistic system would have two complex functions (with a large number of gates) instead of the OR and NAND gates. However, it was decided to analyze the system in Fig. 1 for its simplicity. Furthermore, the system could be made more reliable by having two redundant switches as in the Bus Guardians of the FTMP [Hopkins 78]. The emphasis in this paper is on the calculation of the parameters necessary for the reliability models, not on the design of reliable systems.

The fault model used in this analysis will be the permanent single stuck-at fault model. The effect of temporary failures on the system in Fig. 1 is discussed in [Amer 86b]. The faults are divided into five classes in order to calculate the coverage parameters. Three of these classes correspond to faults in modules X and Y:

1) **Undetectable faults:** Since a fault in lead A (input from Bus-1) for example, will have the same effect on both inputs of the EXCLUSIVE-OR gate, it is undetectable and the system fails (incorrect data on the output bus). An undetectable fault will only cause a system failure if it occurs in a module connected to the bus at the time of the fault.

2) **Detectable faults:** A fault in lead E (input to NAND gate), for example, will only affect one of the inputs of the EXCLUSIVE-OR gate. It will be detected and the switch will permanently disconnect module X from the bus and connect module Y.

3) **Fatal faults:** C s-a-0 (output of primary buffer), for example, can (if C should be 1) force incorrect data on the output bus and the system will immediately fail, irrespective of which module (X or Y) is connected to the bus at the time of the fault.

Two fault classes need to be defined for the recovery mechanism:

4) **Faults causing premature switching:** J s-a-1 (detector output), for example, will cause module X to be permanently disconnected from the bus even though it (module X) is fault-free.

5) **Latent faults:** A latent fault in the detector or switch will not produce an error until a fault occurs in the module they are connected to. J s-a-0 (detector output), for example, will not produce an error until the EXCLUSIVE-OR gate detects an error in module X.

Table 1 Fault Classification

	Group	W[i]	s-a-0	s-a-1	class
Module X	1	20	E,F,G,B2, G1,H,H1, A1,A2,B1	E,F,G,B2, G1,H,H1, A1,A2,B1	detect.
	2	8	A,B,G2,H2	A,B,G2,H2	undetect.
	3	4	C,D	C,D	fatal
Detector	4	1	J		latent
	5	1		J	premature switching
Switch	6	5	K,M,N,M1, M2		latent
	7	16	L,P,R,R1, R2,V,S	K,L,P,R,V, M2,R1,R2,S	fatal
	8	3		N,M,M1	premature switching

W[i] = Number of faults in group i

Latent faults and faults causing premature switching are identical to the "unsafe" and "safe" faults described in [Losq 75]. There are also fatal faults in the switch. If lead R (primary buffer enable signal) is s-a-0, for example, both modules (X and Y) will be disconnected from the bus (when the logic value of the signal on Bus-1 = 0) and the system will fail.

The fault classification is shown in Table 1. Only the faults in module X and the detector and switch connected to it, are shown. The classification of the faults in module Y and the detector and switch connected to it is identical to that shown in Table 1. The faults are divided into eight groups. Each row in Table 1 corresponds to a group and each group corresponds to one of the classes described above. More than one group can belong to the same class. Groups 3 and 7, for example, both consist of fatal faults. The total number of faults in group i is equal to W[i].

3 RELIABILITY CALCULATION USING A SPECIFIC MODEL

Figure 2 shows the Markov model used to calculate the reliability of the system under study. This model takes into account the order in which the different faults occur. It is assumed that the failure rate of any gate or latch in the system is equal to 2z and that the stuck-at-0 and stuck-at-1 faults are equally likely (each with a failure rate of z). Assuming also that all holding times (times spent in each state) are exponentially distributed, the transition rates will be the sum of the appropriate W[i]'s multiplied by z (see Table 1). Solving the Markov model, the system unreliability is obtained as follows [Trivedi 82]:

Unreliability(t) = P {System failure at time t}

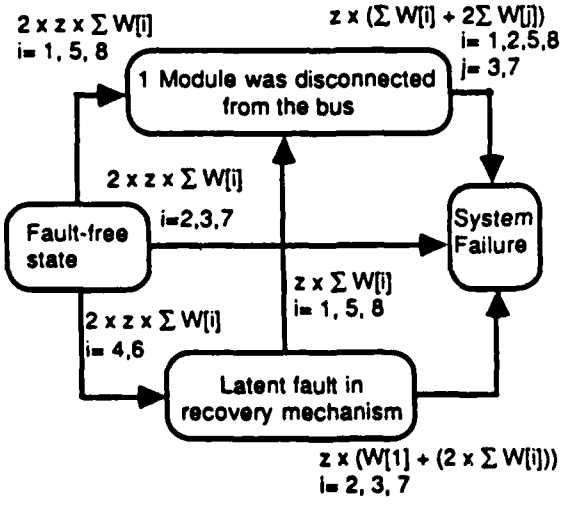


Fig. 2 Reliability model specific to system in Fig. 1

In Fig. 3, the unreliability is plotted (for z = 1 FIT = 1 failure/10^9 hours) along with the unreliability when calculated using CARE III (see Secs. 4 and 5).

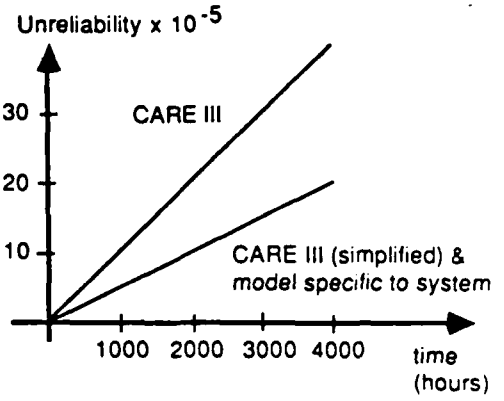


Fig. 3 System Unreliability Calculated Using Model Specific to System, Original and Simplified CARE III Coverage Model

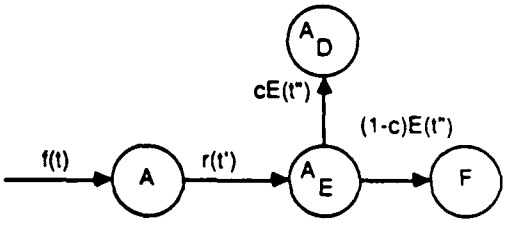


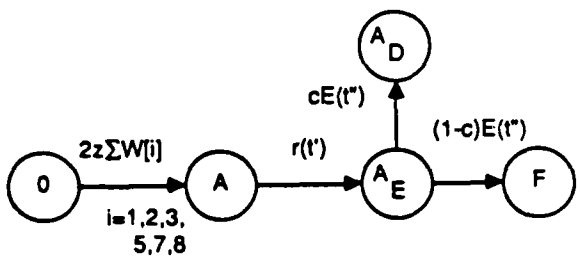
Fig. 4 CARE III Coverage Model

4 RELIABILITY CALCULATION USING CARE III

In this section, the fault classification presented in Sec. 2 is used to determine the parameters for the fault-handling (coverage) model of CARE III. CARE III is a very sophisticated "automated reliability model" [Bridgman 84]. It can be divided into two parts : the aggregate model and the fault-handling (coverage) model. The latter describes the recovery process in detail. More information about CARE III can be found in [Bavuso 84], [Trivedi 81]. Figure 4 shows the single fault-handling model. A fault (with rate f(t)) causes the system being modeled to go to state A. The fault is active but no error exists yet. The fault produces an error (at a rate r(t')) and the system goes from state A to state A_E. If the error is not fatal, the system will go to state A_D (at a rate E(t") and with a conditional probability c). If the error is fatal, the system will go to state F. Both t' and t" are random variables. It is assumed here that they are

distributed exponentially. 1/r(t') is the average time for a fault to produce an error and 1/E(t") is the average time for that error to be detected (or cause a system failure). State A_D indicates that the error was detected; it is assumed in CARE III that the isolation of the error and the recovery from it will always be successful.

Only permanent faults are considered here. The fault-handling model should be able to represent all the faults in the 1-out-of-2 system under study. A latent fault (J s-a-0 for example) will only affect the system after an error in module X. The parameter t' (time for fault to produce an error) will be many orders of magnitude larger than that of a detectable fault in module X. Therefore, the latent faults cannot be handled like the other faults. The system could be divided into two subsystems: Modules X and Y and the recovery mechanism. The double fault-handling model in CARE III [Bavuso 84] can be used to describe the dependency between the faults in the two subsystems.



(0) = Fault-free State

Fig. 5 CARE III with Non-latent Faults

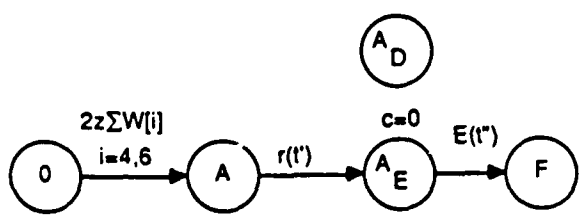


Fig. 6 CARE III with Latent Faults

However, it will be impossible to distinguish the latent faults from the rest of the faults in the recovery mechanism. A solution for this problem is to divide the faults in the system into two types: 1) Latent faults. 2) All other faults in the system (non-latent faults). Hence, the fault-handling model has to be used twice. Dividing the faults in the system into latent and non-latent is the best way of including the faults in the hardware recovery mechanism in the coverage model of CARE III.

In Fig. 5, the fault-handling model is applied to the faults in modules X and Y as well as the non-latent faults in the recovery mechanism. The parameter c (probability that the system can recover from the error) is estimated as the ratio of the faults that are not fatal to the total number of non-latent faults.

$$c = \frac{\text{Sum } W[i] \quad i=1,5,8}{\text{Sum } W[i] \quad i=1,2,3,5,7,8}$$

Since the system is very simple and the clock cycle is many orders of magnitude smaller than the mean time between faults, the parameters r(t') and E(t") are both assumed to be large and constant.

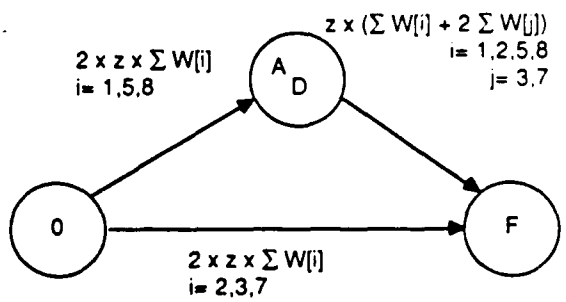


Fig. 7 Simplified coverage model for non-latent faults

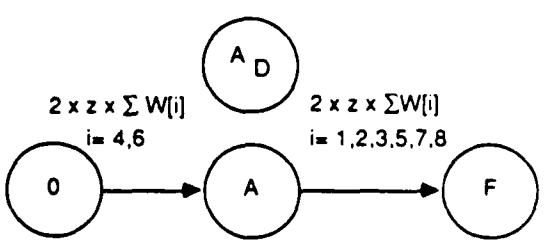


Fig. 8 Simplified coverage model for latent faults

The latent faults in the recovery mechanism are treated separately as shown in Fig. 6. The parameter c, in this case, is equal to zero because any fault in a module whose recovery mechanism is disabled, will lead to a system failure. E(t") is assigned a large constant while r(t') is equal to the sum of the failure rates of the non-latent faults in the system. The reliability is calculated as follows [Trivedi 81]:

Reliability(t) = 1-P{being in state F at time t}

The reliability of the whole system will be the product of the reliabilities calculated from the above two models (Figs. 5 and 6). The unreliability is shown in Fig. 3 for $z = 1$ FIT [Mil 82]. It is conservative when compared to the unreliability obtained from the Markov model specific to the system (Fig. 2).

5 SIMPLIFIED FAULT-HANDLING (COVERAGE) MODEL

In the fault-handling (coverage) model shown in Fig. 5, the parameters r(t') and E(t") were both assumed to be large and constant. r(t') being the rate at which a fault causes an error, it will be many orders of magnitude greater than the failure rate ($z = 1$ FIT $= 1/10^9$ hours). Therefore, it can safely be assumed that the transition between states A and A_E in Fig. 5 is instantaneous. The same argument applies to E(t"). Figure 7 shows a simplified coverage model for the non-latent faults. The parameters r(t') and E(t") are eliminated. Figure 8 shows a simplified coverage model for the latent faults. State A_E is eliminated because errors will instantaneously cause a system failure. Using the same parameters as in Sec. 4, the models shown in Figs. 7 and 8 are solved. It is found that the reliability estimate obtained is identical to the one obtained using the specific model described in Sec. 3 (see Fig. 3). It is conjectured that the difference between the two reliability estimates shown in Fig. 3 is due to the computer roundoff error. In conclusion, dividing the faults in the system into latent and non-latent and including the non-latent faults in the recovery mechanism with the faults in modules X and Y produces an excellent estimate of the system reliability.

6 SUMMARY and CONCLUSIONS

A very simple redundant system was designed. It consists of two identical modules that are interchangeably connected to the system bus. The recovery mechanism consists of a detector and a switch for each module. The faults were divided into five classes assuming a single stuck-at fault model. The five fault classes were then used to determine the coverage parameters for the fault-handling model in CARE III. Even though only one type of fault (permanent) was modeled, the fault-handling (coverage) model had to be used twice to account for the latent faults in the recovery mechanism. Each time the model was used, a different set of coverage parameters was calculated.

A reliability model specific to the system was also developed and the results were compared to those obtained with the models in CARE III. It was found that CARE III gives a conservative estimate of the reliability of the system.

Finally, the fault-handling (coverage) model of CARE III was simplified. It was then used to calculate the reliability of the system and it was found that the estimate obtained was identical to the one obtained from the Markov model specific to the system. It was conjectured that the computer roundoff error was responsible for the conservative estimate obtained from the original (unmodified) coverage model.

ACKNOWLEDGMENTS

The authors thank M. Cortes, A. Ersoz and K. Wagner for their help and advice on text preparation. This work was supported in part by the National Science Foundation under Contract Number NSF DCR8200129 and by the Egyptian government.

REFERENCES

[Amer 86a] H.H. Amer and E.J. McCluskey, "Calculation of the Coverage Parameter for the Reliability Modeling of Fault-Tolerant Computer Systems," **Proceedings of the International Symposium on Circuits and Systems** ISCAS-86, San Jose, CA, May 1986.

[Amer 86b] H.H. Amer and E.J. McCluskey, "Inadequacy of Dynamic Recovery Mechanisms in the Presence of Temporary Failures," CRC **Technical Report**, Stanford Un.

[Arnold 72] T.F. Arnold, "The Concept of Coverage and its Effect on the Reliability Model of a Repairable System," **Proc. FTCS-2**, Newton, MA, 1972, pp. 200-204.

[Bavuso 84] S.J. Bavuso, "A User's View of CARE III," **Proceedings of the 1984 Annual Reliability and Maintainability Symposium**, pp. 417-422, Jan. 1984.

[Borgerson 75] B. Borgerson and R. Freitas, "A Reliability Model for Gracefully Degrading and Stand-by Sparing Systems," **IEEE Trans. Comput.**, May 1975.

[Bouricius 71] W. Bouricius, W. Carter, D. Jessep, P. Schneider and A. Wadia, "Reliability Modeling for Fault-Tolerant Computers," **IEEE Trans. Comput.**, Nov. 1971.

[Bridgman 84] M. Bridgman and W. Ness, "Automated Ultrareliability Models: A Review," **Proceedings of the 1984 Annual Reliability and Maintainability Symposium**, Jan. 1984, pp. 396-402.

[Hopkins 78] A. Hopkins, T. Basil Smith and J. Lala, "FTMP- A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," **Proc. IEEE**, Vol. 66, No. 10, Oct. 1978, pp. 1221-1239.

[Losq 75] J. Losq, "Influence of Fault Detection and Switching Mechanisms on the Reliability of Stand-By Systems," **Proc. FTCS-5**, Paris, France, 1975, pp. 81-86.

[Mil 82] U.S. Department of Defense, "Military Handbook: Reliability Prediction of Electronic Equipment", **MIL-HDBK-217D**, Jan. 1982.

[Siewiorek 82] D.P. Siewiorek and R.S. Swarz, "**The Theory and Practice of Reliable System Design**," Digital Press, Bedford, MA, 1982.

[Trivedi 81] K.S. Trivedi and R.M. Geist, "A Tutorial on the CARE III Approach to Reliability Modeling," **NASA Contractor Report** 3488, Dec. 1981.

[Trivedi 82] K.S. Trivedi, "**Probability and Statistics with Reliability, Queuing and Computer Science Applications**", Prentice-Hall, NJ, 1982.

A VLSI CMOS CIRCUIT DESIGN TECHNIQUE TO AID TEST GENERATION

Dick L. Liu and Edward J. McCluskey

Center for Reliable Computing
Computer Systems Laboratory
Stanford University, Stanford, CA 94305-4055

ABSTRACT

This paper describes a new design technique for fully-testable CMOS combinational circuits and a 3-pattern test scheme to detect switch-level (stuck-open and stuck-on) faults. A fully-testable combinational circuit is implemented with specially designed gates that have no undetectable stuck-on faults. Switch-level faults in this type of combinational circuit can be detected with a 3-pattern test scheme. These 3-pattern tests are easy to generate with a gate-level automatic test pattern generator (ATPG) that offers better performance than a switch-level ATPG.

1 INTRODUCTION

CMOS circuits possess certain unique failure modes that cannot be detected by a stuck-at fault test set. These failure modes are better modeled at the transistor circuit or the switch level as FET stuck-open faults and FET stuck-on faults [Wadsack 78].

A *stuck-open* fault requires a sequence of two consecutive patterns to guarantee detection. However, these 2-pattern tests can potentially be invalidated by stray circuit delays [Reddy 83]. The algorithm to generate valid robust 2-pattern tests is computationally complex. It has been reported that a switch-level, stuck-open fault ATPG may require an order of magnitude more CPU time to generate tests than a gate-level, stuck-at fault ATPG, and it can only achieve stuck-open fault coverage of up to 70% on large ICs [Moritz 86].

A *stuck-on* fault presents a different testing problem. Consider the logic gate shown in Fig. 1. The test pattern that should detect FET N2 stuck on is (1, 0, 1, 0). However, this input combination creates a short between Vdd and ground within the faulty gate. If the circuit is P-dominant (i.e. the transconductance of every PFET is much larger than every NFET), then the voltage level at output node Z will be close to 5 volts. In this case, the faulty gate output will be recognized as the same as the fault-free gate output and the fault is not detected. These undetectable faults are difficult to identify and should be removed from the fault list. [Lusky 85] reported that the presence of undetectable stuck-on faults in a CMOS VLSI circuit will substantially slow down switch-level ATPG and fault simulation programs.

Due to the difficulty of testing for switch-level faults in conventional CMOS circuits, design for testability (DFT) techniques were proposed to offer circuit structures which have better testability for switch-level faults. [Jha 85; McCluskey 81; Reddy 83] proposed testable design methods to facilitate the detection of stuck-open faults in a single logic gate. Stuck-on fault testability was not addressed. [Zasio 85] described a layout method to decrease the probability of stuck-open fault occurrence in gate array chips, but his method cannot be generalized to custom VLSI circuits. [Brzozowski 85] presented a testable gate design method for both stuck-open and stuck-on faults. However, the method cannot be easily extended to a combinational circuit consisting of interconnections of testable gates. [Malaiya 82] proposed a current monitoring technique to detect stuck-on faults. Nonetheless, this technique will slow down functional test substantially and prohibits the use of dynamic circuit structures that draw steady-state current.

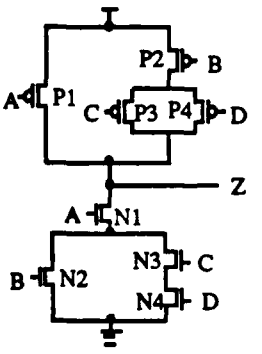


Figure 1. An FCMOS gate example.

This paper describes a new CMOS circuit design technique for switch-level testability. The technique considers both the stuck-open fault and the stuck-on fault. It creates testable combinational circuits for which a conventional gate-level ATPG can generate tests.

2 TESTABLE DESIGN TECHNIQUES

A CMOS combinational circuit consists of various types of logic gates: primitive gates (e.g. inverter, NAND, NOR) and complex gates (e.g. AOI gate, OAI gate). Most logic gates have the following properties: (1) each logic gate input is connected to the transistor gate terminals of both a PFET and an NFET, and (2) the pull-up network provides conduction paths for all input combinations for which the output node is one; the pull-down network provides conduction paths for all input combinations for which the output node is zero. These gates will be referred to as *Fully Complementary MOS (FCMOS) gates* in this paper. The logic gate of Fig. 1 is an example of FCMOS gates. Gates that are not FCMOS gates are often used as I/O buffers and in busses, and examples of these gates can be found in [Reddy 84].

2.1 Circuit design for stuck-open fault testability

This subsection summarizes a circuit structure and its test scheme presented in [Liu 86a].

Detecting a stuck-open fault in an FCMOS gate requires two patterns. The first pattern *(initializing input)* is applied to charge or discharge the gate output, and the second pattern *(test input)* is applied to change the value of the output node through the faulty FET. A simplified 2-

pattern test which uses either an all-one or an all-zero pattern as the first input pattern is easier to generate than other 2-pattern tests. These simplified patterns were proved to be valid under stray circuit delays.

To apply simplified patterns to an embedded FCMOS gate, the combinational circuit must be implemented as shown in Fig. 2. In this figure, an inverting buffer is added to every FCMOS gate which drives other FCMOS gate(s). If all the primary input literals (x and x') are set to zero, then all of its FCMOS gate inputs are initialized to zero; if all the primary inputs are set to one, then all of its FCMOS gate inputs are initialized to one. Therefore, it is straightforward to generate the first input pattern for simplified 2-pattern testing of any embedded FCMOS gate.

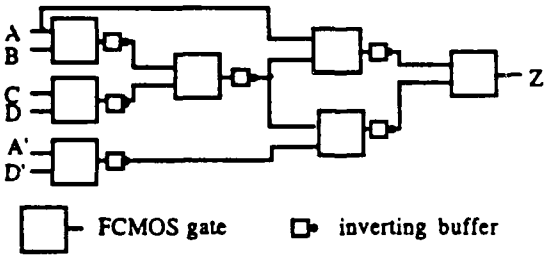


Figure 2. Block diagram of a testable circuit.

2.2 Circuit design for stuck-on fault testability

The following definitions describe a logic gate structure in which every stuck-on fault in the pull-up and the pull-down networks is detectable.

Definition: A *Stuck-On Fault Testable (SOFT) gate* is constructed by adding two FETs (*blocking FETs*) and two input lines to an FCMOS gate, Fig. 3(a). FET Pb connects Vdd to the pull-up network and FET Nb connects ground to the pull-down network. Input line Cp' controls FET Pb and input line Cn controls FET Nb.

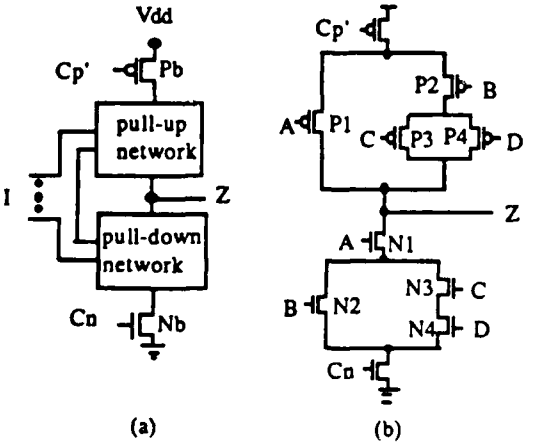


Figure 3. A stuck-on fault testable gate.
(a) Block diagram. (b) An example.

Definition: The input lines to the pull-up and pull-down network of a SOFT gate are called *functional inputs*. Input lines Cp' and Cn of a SOFT gate are called *control inputs*. During normal operation, control input Cp' is set to zero and Cn is set to one.

With the addition of blocking FETs, every stuck-on fault in the pull-down and pull-up network of a SOFT gate is detectable by a pair of test patterns. (Stuck-on faults in blocking FETs require a special testing technique, and will be discussed in Section 3) First, an input pattern is applied to the SOFT gate to provoke the stuck-on fault and propagate its faulty effect (an intermediate voltage level) to the output node. Second, an input pattern is applied to de-activate one of the blocking FETs which breaks the short between Vdd and ground. The output node will then be charged or discharged to a faulty value through the stuck-on FET and the fault is thus detected.

Example 1: If FET N2 in Fig. 3(b) is stuck-on, find a pair of test patterns for this fault.

The following input patterns can detect the stuck-on fault in FET N2:

$$\{(Cp', Cn, A, B, C, D)\} = \{(0,1,1,0,1,0), (1,1,1,0,1,0)\}.$$

The first input pattern creates a short in the faulty SOFT gate but not in the fault-free gate, and sets the output node to an intermediate voltage level (x). The second pattern turns FET Pb off and discharges the faulty gate output to zero. However, the fault-free gate output node is at a high-impedance state and will remain at one. Therefore, the stuck-on fault is detected.

A combinational circuit can be constructed by interconnecting SOFT gates. However, this presents a problem when testing for stuck-on faults. In Fig. 4, assume a stuck-on fault exists in the pull-down network of SOFT gate A. Applying a pair of test patterns to this logic gate will force the output node U to change from an intermediate voltage (x) to zero because the second pattern will set Cp' to one. However, the transition at node U (x to zero) cannot change the logic value at the output node V because the pull-up network of SOFT gate B is also disconnected from Vdd. Therefore, the faulty effect in gate A will not be propagated through gate B to an observable output.

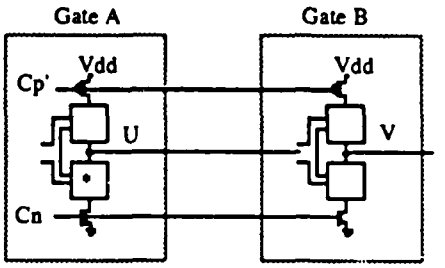


Figure 4. A cascade of SOFT gates.

An obvious solution to this error propagation problem is to use two sets of Cp' and Cn signals to control SOFT gates in an alternating fashion. However, this solution increases wiring complexity and complicates the control circuitry. A better solution is to add inverting buffers between SOFT gates so that the combinational circuit conforms to the structure shown in Fig. 2 except that FCMOS gates are replaced by SOFT gates. Notice these inverting buffers do not include blocking FETs. Due to the buffer, gate B will receive an input transition from x to one instead of x to zero. When the second input pattern is applied to the CUT, the control input Cn is set to one so that every pull-down network is connected to ground. An x-to-one transition at the input

of gate B can propagate to its output. Thus, a stuck-on fault in this circuit structure can be detected with a pair of patterns.

2.3 Fully testable circuit structure

In summary, a fully testable combinational circuit can be defined as follows:

Definition: A *fully testable combinational circuit* is a multi-level combinational circuit which consists of SOFT gates and inverting buffers. An inverting buffer is placed at the output of every SOFT gate which drives another SOFT gate(s).

Two combinational circuits were examined to evaluate the impact of the proposed design technique on circuit area and speed. These were an 8-to-1 multiplexer (MUX81H) and a 4-bit adder (FA4) [LSI 85]. The average area overhead was found to be approximately 25%. This figure represents a worst case estimate because no attempt was made to reduce the overhead by sharing blocking FETs or by minimizing device sizes. The performance impact was determined by SPICE simulations using parameters from Stanford University's Center for Integrated Systems CMOS process. The average critical path delay of the two testable designs was increased by 15%.

2.4 Three-pattern testing of switch-level faults

This subsection discusses how to combine the test patterns for a stuck-open fault and a stuck-on fault so that they can be tested by a sequence of three test patterns. Furthermore, it will be shown that these 3-pattern tests can be derived from a gate-level, stuck-at fault ATPG.

Test patterns for detecting switch-level faults in a SOFT gate are shown in Table 1. In this table, both Ta and Tb are input patterns for provoking the stuck-on faults in the SOFT gate; Tc and Td are input patterns for detecting stuck-open faults. For stuck-on fault patterns, the second input pattern is identical to the first input pattern except for the control inputs. The second pattern must turn off one of the blocking FETs so as to break the short in the faulty logic gate. The location of the stuck-on fault within the SOFT gate determines the input pattern for control inputs. For example, if a stuck-on fault is in the pull-down network, the second pattern is required to set input Cp' to one so as to isolate the pull-up network from Vdd. For stuck-open fault patterns, the initializing input pattern is similarly determined by the location of the fault within a SOFT gate. For example, if the stuck-open fault is in the pull-down network, an all-zero pattern is used for the functional inputs to charge the output node Z.

Next, we introduce the concept of duality within a SOFT gate.

Definition: The PFET Pi and NFET Ni whose transistor-gate terminals are controlled by the same input line i of a SOFT gate are called the *associate FETs* of input line i. FET Pi is called the *dual FET* of Ni and vice versa.

Definition: Within a SOFT gate, if a PFET Pi has a stuck-open fault, its *dual fault* is defined as a stuck-on fault in its dual FET Ni; similarly, the dual fault of a stuck-on PFET Pi is defined as a stuck-open fault in the dual FET Ni.

The following lemma is due to [Chiang 83].

Lemma 1: (Duality) For an FCMOS gate, the test input pattern for a stuck-open fault will detect a stuck-on fault in its dual FET if that stuck-on fault is detectable.

Since every stuck-on fault in a SOFT gate is detectable, the above lemma can be used to combine the test set for stuck-open faults with that for stuck-on faults. This leads to the following corollary:

Table 1. A summary of test patterns for a SOFT gate.

Test for a stuck-on fault.									
Fault in the PD-network					Fault in the PU-network				
Cp'	Cn	I	Z	Z*	Cp'	Cn	I	Z	Z*
0	1	Ta	1	x	0	1	Tb	0	x
1	1	Ta	1*	0	0	0	Tb	0*	1
Test for a stuck-open fault.									
Fault in the PD-network					Fault in the PU-network				
Cp'	Cn	I	Z	Z*	Cp'	Cn	I	Z	Z*
0	1	all-0	1	1	0	1	all-1	0	0
0	1	Tc	0	1*	0	1	Td	1	0*

Z: fault-free output — Z*: faulty output
x: intermediate voltage — 0*: high-impedance 0
PU-network: pull-up network — 1*: high-impedance 1
PD-network: pull-down network

Corollary 1: (Three-pattern test) In a SOFT gate, every stuck-open fault and its dual stuck-on fault can be tested with a 3-pattern test.

A 3-pattern test for a SOFT gate is shown in Table 2. TA and TB are test input patterns for stuck-open faults. The first two (T1 and T2) patterns will detect a stuck-open fault and the last two (T2 and T3) patterns will detect the dual stuck-on fault. Only the T2 pattern needs to be generated, the T1 and T3 patterns are then easily determined. As a matter of fact, the T2 pattern can be generated by a gate-level, stuck-at fault ATPG.

Table 2. 3-pattern tests for a SOFT gate.

PU-network / PD-network	stuck-open / stuck-on					stuck-on / stuck-open				
	Cp'	Cn	I	Z	Z*	Cp'	Cn	I	Z	Z*
T1	0	1	all-1	0	0	0	1	all-0	1	1
T2	0	1	TA	1	0*	0	1	TB	0	1*
T3	1	1	TA	1	0	0	0	TB	0	1

For the purpose of test generation, a CMOS logic gate can be represented either at the switch level or at the logic gate level. For example, Fig. 5 shows two different representations of a SOFT gate. Notice that the blocking FETs of a SOFT gate are ignored when deriving the equivalent gate-level representation. Each stuck-at fault on an input line in Fig. 5(b) corresponds to two switch-level faults in Fig. 5(a). Nonetheless, there exists a correspondence between the test patterns for a stuck-at fault and a pair of switch-level faults.

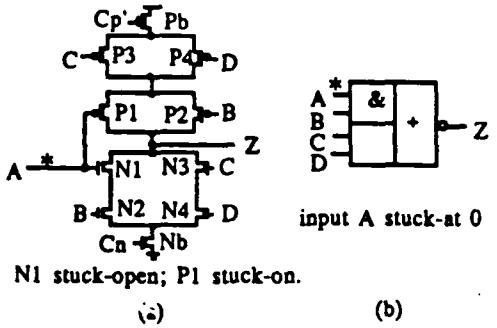


Figure 5. A SOFT gate. (a) Switch-level representation. (b) Gate-level representation.

Lemma 2: For a SOFT gate, the test pattern that detects an input line i stuck-at fault is a T2 pattern for the switch-level faults in the FETs associated with line i.

Proof: Assume the SOFT gate implements a Boolean function $f(x_1, x_2, ..., x_n)$. If a stuck-at zero fault exists on input line x_i of the gate-level representation, this fault corresponds to a stuck-open fault in the pull-down network and a stuck-on fault in the pull-up network. The pull-down network can be expressed by a transmission $Mn(x_1, x_2, ..., x_n)$ that is equivalent to f. The T2 pattern for the stuck-open fault can be found by computing the Boolean difference of Mn, i.e. dMn/dx_i [McCluskey 86]. From the property of Boolean difference,

$$dMn/dx_i = df/dx_i = df/dx_i,$$

where the test pattern for a stuck-at fault on line x_i can be found from df/dx_i. Therefore, the stuck-at fault pattern for line x_i is the same as the T2 pattern for its associated switch-level faults.

If input line x_i has a stuck-at-one fault, it will correspond to a stuck-open fault in the pull-up network and a stuck-on fault in the pull-down network. The pull-up network can be expressed as a transmission $Mp(x'_1, x'_2, ..., x'_n)$ that is equivalent to f. However, for a fully complementary gate, the pull-up network is a dual of the pull-down network. Therefore

$$Mp(x'_1, x'_2, ..., x'_n)=[D(Mp(x_1, x_2, ..., x_n))]' =Mn',$$

where D(M) stands for the dual of transmission M.
To compute the T2 pattern for the stuck-open fault, we need to find the Boolean difference of Mp. From the property of Boolean difference,

$$dMp(x'_1, x'_2, ..., x'_n)/dx'_i = dMn'/dx'_i = df/dx'_i = df/dx_i.$$

Therefore, a test pattern for the x_i stuck-at one fault is the same T2 pattern for its associated switch-level faults.

This lemma is true even if an input line controls more than two FETs. Detailed discussion is presented in [Liu 86b].

Q.E.D.

Based upon the result of this lemma, a gate-level, stuck-at fault ATPG can be used to generate the T2 test patterns for switch-level faults in a SOFT gate. The ability to use a gate-level ATPG will greatly speed up the process of test generation for CMOS circuits.

3 TEST SCHEMES FOR ADDITIONAL CIRCUITRY

The additional test circuitry for a fully testable combinational circuit consists of inverting buffers and blocking FETs. This circuitry must be checked before 3-pattern tests can be used to detect switch-level faults in the remaining functional circuitry. Detecting stuck-open faults in the additional circuitry is straightforward and requires no additional test patterns. However, detecting stuck-on faults in this circuitry is very difficult with only functional test patterns.

It has been shown in [Liu 86] that additional test patterns are not necessary for detecting stuck-open faults in the inverting buffers. A stuck-open fault in the blocking FET Nb (Fig. 6(a)) is equivalent to a stuck-open fault in the pull-down network. Because it is necessary to set input Cn to one and establish a conduction path in the pull-down network to provoke the stuck-open fault in Nb, this input pattern can also detect any stuck-open fault in the conduction path. Therefore, if we generate a complete test for every stuck-open fault in the pull-down network, this test set will simultaneously detect the stuck-open fault in FET Nb. Similarly, a stuck-open fault in FET Pb is equivalent to a stuck-open fault in the pull-up network.

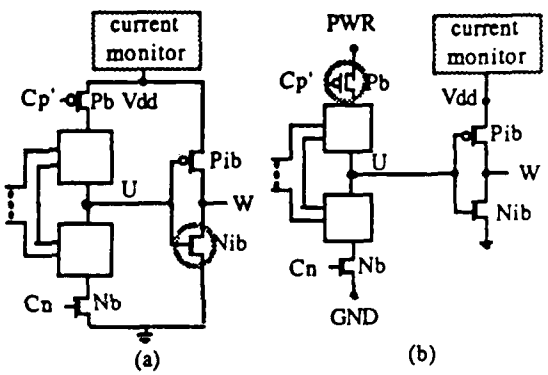


Figure 6. Test schemes for additional circuitry. (a) Inverting buffers. (b) Blocking FETs.

The scheme for testing stuck-on faults in an inverting buffer is shown in Fig. 6(a). If FET Nib is stuck on, we need to produce a zero at the inverting buffer input node U so that this faulty inverter will draw a large supply current. Node U can easily be set to zero by applying an all-one pattern at the primary inputs of the CUT. Notice that the NFET stuck-on fault in every inverting buffer can be tested at the same time by setting the control input Cn to one, Cp' to zero and every functional primary input to one. The PFET stuck-on fault in every inverting buffer can be tested similarly.

This current monitoring scheme can also be extended to detect a stuck-on fault in blocking FETs as shown in Fig. 6(b). Assuming that two sets of power and ground lines are used: one for the SOFT gates and the other for the inverting buffers. If FET Pb is stuck-on, we apply a primary input pattern which sets Cn to one, Cp' to zero and the remaining functional inputs to zero. Node U will be set to one. Then control input Cp' is changed to one and the voltage level at the PWR line is decreased from 5 volts to 2.5 volts. If FET Pb is stuck-on, the voltage level at node U will follow the voltage decrease at the PWR line. This intermediate voltage will cause the

succeeding inverting buffer to draw a very large current from its power supply. This large leakage current can be detected by a current monitor. Similarly, by stressing the voltage level at the GND line, a stuck-on fault at any blocking FET Nb can be detected.

4 SUMMARY AND CONCLUSIONS

Many ATPGs use a switch-level circuit model to accommodate CMOS stuck-open and stuck-on faults. However, the model's effectiveness is limited. As an alternative, this paper proposes a design technique to enhance the testability of CMOS combinational circuits.

This technique consists of implementing combinational circuits with specially designed gates (SOFT gates) and inserting an inverting buffer after every SOFT gate with a non-primary output. A 3-pattern test scheme can then be used to detect the switch-level faults in this circuit structure. These 3-pattern tests can be generated by a gate-level ATPG for stuck-at faults and they cannot be invalidated by stray circuit delays.

ACKNOWLEDGEMENTS

The authors wish to thank Prof. John Hennessy and John Acken for their encouragement and assistance. The comments provided by Hassanein Amer, Jon Udell and Ken Wagner are also gratefully achnowledged. This work was supported in part by the Defense Advanced Research Projects Agency (Darpa) under Grant No. MDA. 903-83-C-0335.

REFERENCES

[Brzozowski 85] Brzozowski, J.A., "Testability of CMOS Cells," Tech. Report No. CS-85-31, Dept. of Computer Science, University of Waterloo, Waterloo, Ont., Canada, Sept. 1985.

[Chiang 83] Chiang, K.W., and Z.G. Vranesic, "On Fault Detection in CMOS Logic Networks," Proc. 20th Design Automation Conf., pp.50-56, 1983.

[Jha 85] Jha, N.K. and J.A. Abraham, "Design of Testable CMOS Logic Circuits under Arbitrary delays," IEEE Trans. Computer-aided Design, Vol. CAD-4, No.3, pp.264-269, July 1985.

[Liu 86a] Liu, D., and E.J. McCluskey, "Design of CMOS VLSI Circuits for Testability," Proc. 1986 IEEE Custom Integrated Circuits Conf., pp.421-424, 1986.

[Liu 86b] Liu, D., and E.J. McCluskey, "Design CMOS Combinational Circuits for Switch-level Testability," Tech. Report, Center for Reliable Computing, Stanford University, Stanford, CA, to be published in 1986.

[LSI 85] LSI Logic Corp., CMOS Macrocell Manual, Milpitas, CA, July 1985.

[Lusky 85] Lusky, S., and T. Sridhar, "Detectable CMOS Faults in Switch-level Simulation," Proc. 1985 Int. Test Conf., pp. 875-883, 1985.

[McCluskey 81] McCluskey,E.J. and S. Bozorgui-Nesbat, "Design for Autonomous Test," IEEE Trans. Computer, Vol. C-30, pp.866-875, Nov. 1981.

[McCluskey 86] McCluskey, E.J., Logic Design Principles - with emphasis on Testable Semicustom Circuits, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[Malaiya 82] Malaiya, Y.K., and S.Y.H. Su, "A New Fault Model and Testing Technique for CMOS Devices," Proc. 1982 Int. Test Conf., pp. 25-34, 1982.

[Moritz 86] Moritz, P.S., and L.M. Thorsen, "CMOS Circuit Testability," IEEE Journal of Solid-State Circuits, Vol.SC-21, No.2, pp. 306-309, April 1986.

[Reddy 83] Reddy, S.M., M.K. Reddy and J.G. Kuhl, "On testable design for CMOS logic circuits," Proc. 1983 Int. Test Conf., Philadelphia, PA, pp.435-445, 1983.

[Reddy 84] Reddy, S.M., V.D. Agrawal and S.K. Jain, "A Gate Level Model for CMOS Combinational Logic Circuits with Application to Fault Detection," Proc. 21st Design Automation Conf., pp.504-509, 1984.

[Wadsack 78] Wadsack, R.L., "Fault modeling and logic simulation of CMOS and MOS Integrated Circuits," Bell System Technical Journal, Vol.57, No.5, pp.1449-1473, May-June 1978.

[Zasio 85] Zasio, J.J., "Non Stuck Fault Testing of CMOS VLSI," CompCon, Spring, pp. 388-391, 1985.

END

DATE

FILMED

FEB.

1988